VŠB – Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Absolvování odborné praxe

Individual professional practice in the company

2016 Daniel Hrtús

VŠB - Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Zadání bakalářské práce

Student: **Daniel Hrtús**

Studijní program: B2647 Informační a komunikační technologie

Studijní obor: 2612R025 Informatika a výpočetní technika

Téma: Absolvování individuální odborné praxe
Individual Professional Practice in the Company

Jazyk vypracování: čeština

Zásady pro vypracování:

1. Student vykoná individuální praxi ve firmě: ComProMis, s.r.o.
2. Struktura závěrečné zprávy:
   a) Popis odborného zaměření firmy, u které student vykonal odbornou praxi a popis pracovního zařazení studenta.
   b) Seznam úkolů zadaných studentovi v průběhu odborné praxe s vyjádřením jejich časové náročnosti.
   c) Zvolený postup řešení zadaných úkolů.
   d) Teoretické a praktické znalosti a dovednosti získané v průběhu studia uplatněné studentem v průběhu odborné praxe.
   e) Znalosti či dovednosti scházející studentovi v průběhu odborné praxe.
   f) Dosažené výsledky v průběhu odborné praxe a její celkové zhodnocení.

Seznam doporučené odborné literatury:

Podle pokynů konzultanta, který vede odbornou praxi studenta.

Formální náležitosti a rozsah bakalářské práce stanoví pokyny pro vypracování zveřejněné na webových stránkách fakulty.

Vedoucí bakalářské práce: **Ing. Petr Gajdoš, Ph.D.**

Konzultant bakalářské práce: Ing. Roman Potoczný

Datum zadání: 01.09.2015

Datum odevzdání: 29.04.2016

VYSOKÁ ŠKOLA BÁŇSKÁ-TECHNICKÁ UNIVERZITA Fakulta elektrotechniky a informatiky 1D16 OSTRAVA

doc. Dr. Ing. Eduard Sojka
vedoucí katedry

prof. RNDr. Václav Snášel, CSc.
děkan fakulty

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

V Ostravě 29. dubna 2016 .....

Souhlasím se zveřejněním této bakalářské práce dle požadavků čl. 26, odst. 9 Studijního a zkušebního řádu pro studium v bakalářských programech VŠB-TU Ostrava.

V Ostravě 29. dubna 2016

.....

ComPromis s.r.o.
IČ: 64618838 DIČ: CZ64618838
Vratimovská 624/11, 718 00 Ostrava-Kunčičky

Rád bych poděkoval firmě ComProMiS s.r.o. za možnost absolvování odborné praxe, především Ing. Romanovi Potocznému za konzultace a vedení mé praxe. Poděkování také patří vedoucímu práce Ing. Petru Gajdošovi, Ph.D. za čas, který mi věnoval při tvorbě této bakalářské práce.

Abstrakt

Tato bakalářská práce popisuje průběh absolvování odborné praxe ve firmě ComProMiS s.r.o. Nejprve uvedu zaměření firmy a mou pozici, na které jsem praxi vykonával. Poté bude následovat popis jednotlivých projektů a úkolů, se kterými jsem se ve firmě setkal. Cílem praxe byla analýza a podílení se na nových, či stávajících projektech, konkrétně se jednalo o tvorbu nových webových stránek pro firmu, spolupráce na aplikaci TOrders a vytváření reportů, a nakonec vytvoření publikačního systému. Na závěr zhodnotím dosažené výsledky a přínos praxe.

Klíčová slova: ComProMiS s.r.o., bakalářská práce, ASP.NET, MVC, HTML, CSS, Javascript, MySQL, Entity Framework, Subversion, report

Abstract

This bachelor thesis describes the process of absolving professional practice in ComProMiS s.r.o. company. First, I will describe specialization of company and my position, which I worked on. In the following part will be description of the various projects and tasks, which I was assigned to work on. The aim of the practice was the analysis and collaboration of new or existing projects, specifically it was the creation of a new website for the company, cooperation on the application TOrders and reporting, and ultimately the creation of the publishing system. In conclusion, I will evaluate the results and benefits of the practice.

Key Words: ComProMiS s.r.o., bachelor thesis, ASP.NET, MVC, HTML, CSS, Javascript, MySQL, Entity Framework, Subversion, report

Obsah

Seznam použitých zkratek a symbolů

API	–	Application Programming Interface
CSS	–	Cascading Style Sheets
EF	–	Entity Framework
HTML	–	Hypertext Markup Language
IIS	–	Internet Information Services
JSON	–	Javascript Object Notation
MVC	–	Model-View-Controller
ORM	–	Object-Relational Mapping
PDF	–	Portable Document Format
SVN	–	Subversion
URL	–	Uniform Resource Locator

Seznam obrázků

Seznam výpisů zdrojového kódu

1 Úvod

V této práci popíši průběh a přínos mé bakalářské práce. Bakalářská práce proběhla formou vykonání odborné praxe ve firmě. Tato forma pro mne byla přitažlivější z důvodu získání praktických zkušeností a práce na zadané téma ve specializované firmě. Pro vykonání odborné praxe jsem oslovil firmu ComProMiS s.r.o. Tato firma se zaměřuje především na vývoj aplikací pro platformu *.NET*, ale také správou UNIX systémů a jiné. Právě vývoj pro platformu *.NET* byla má oblast zájmu. Před nastoupením na praxi jsem měl základní znalosti programování v jazyce C# pro platformu *.NET* ze školních předmětů Programovací jazyky II, Architektura .NET a Vývoj Informačních Systémů. Během praxe ve firmě jsem pracoval na různých projektech a úkolech, které mi byly zadány.

V první kapitole práce více přiblížím firmu ComProMiS s.r.o. a její zaměření. Další část je věnována již zmíněným projektům. Prvním projektem byla tvorba nových webových stránek pro firmu. Důležité body tohoto projektu byly responzivní design a jazykové verze jak pro češtinu, tak angličtinu. Druhým projektem byla spolupráce na firemní aplikaci *TOrders*. Seznámil jsem se s technologií reportů a prací s verzovacím systémem *Subversion*. Posledním projektem je tvorba publikačního systému. Díky tomuto projektu jsem získal zkušenosti s architekturou MVC a *Entity Frameworkem*. Na závěr práce zhodnotím dosažené výsledky, získané jak praktické, tak teoretické znalosti a celkový přínos praxe.

2 Firma ComProMiS s.r.o

Společnost ComProMiS, s.r.o. se specializuje na zakázkovou tvorbu software a údržbu specializovaných informačních systémů. Má dlouhodobé zkušenosti s tvorbou informačních systémů, klient-server aplikací, intranetových aplikací, webových stránek a aplikací pro mobilní telefony.

Jsou partnerem společnosti Software AG a jako jedni z mála v České republice a na Slovensku umí tvořit software na operačním systému Unix (HP-UX, AIX) a na bázi databáze Adabas a vývojového prostředí Natural.

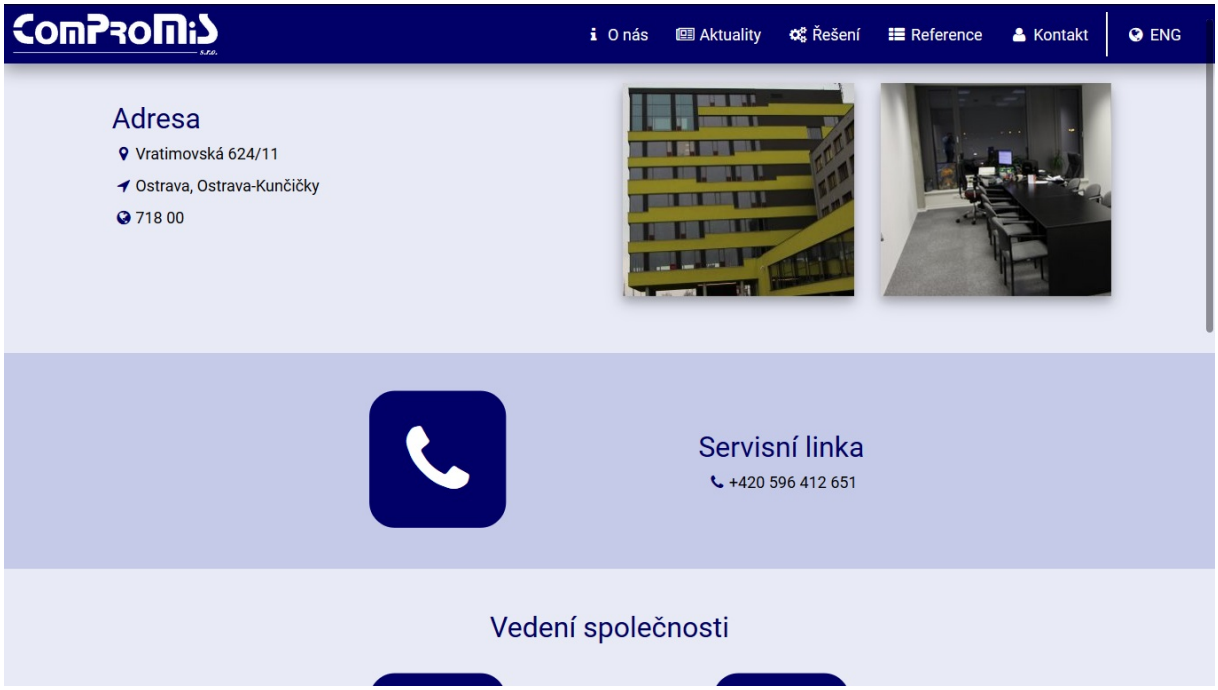
Přestože výše uvedená schopnost představuje v dnes již téměř dvacetileté historii, zásadní historický milník, nyní se soustředí převážně na vývoj v prostředí operačního systému Windows a v moderních technologiích na bázi .NET Framework, jako jsou ASP.NET, MVC, C# a Java v databázových prostředích Microsoft SQL Server nebo Oracle. Většina jejich aplikací cílí do prostředí B2B a B2C, tedy do prostředí podpory nákupu, zpracování zakázek a výměny informací mezi centrálami a pobočkami, včetně podpory nákupu přes moderní portály. Nemalou část aktivit tvoří i tvorba aplikací pro podporu servisních aktivit. [1]

Mé pracovní zařazení ve firmě bylo na pozici analytik a vývojář projektů pro platformu .NET. Jako hlavní náplní mé praxe tedy bylo programování v jazyce C#, tvorba webových stránek za použití HTML a CSS a architektury MVC. Jako vývojové prostředí jsem používal Visual Studio 2015 od společnosti Microsoft.

3 Webové stránky firmy

3.1 Zadání

Prvním úkolem ve firmě bylo vytvoření nových webových stránek pro firmu ComProMiS s.r.o. Nejdříve proběhla analýza požadavků firmy na stránky, rozhodnutí pracovního postupu, použitých technologií a následně samotné vypracování. Stránky měly být jednoduché a podat jasné informace o firmě. Od začátku bylo třeba počítat jak s českou, tak anglickou verzí webu. Mezi dalšími hlavními požadavky byl responzivní design a později také systém pro načítání novinek z databáze. Styl a barevné provedení webu je spjato s firemními barvami. Průběh práce a úpravy aplikace byly průběžně konzultovány s vedoucím praxe. Ukázka aplikace je na obrázku 1.



Obrázek 1: Ukázka aplikace

3.2 Použité technologie

3.2.1 ASP .NET Framework

Pro vývoj webu byl použit *ASP.NET Framework*. *ASP.NET* nabízí široké možnosti a podporu při tvorbě různých webových aplikací. Tento framework byl využit především kvůli zkušenostem ve firmě a taky jednoduchému nasazení na firemní server. Mezi tři základní typy *ASP.NET* aplikace patří *Web Pages*, *Web Forms* a *MVC*, právě poslední zmíněný typ byl použit pro vývoj této aplikace. MVC architektura je složena ze tří hlavních částí. Model je objekt, který uchovává data. View je HTML šablona do které může být zaslán model s daty. Controller je třída, která

řídí logiku aplikace, upravuje model a zobrazuje view. Šablony jsou klasické HTML stránky rozšířené o jazyk *Razor*. Ten umožňuje vkládat C# kód do view pro práci s modelem.

3.2.2 Entity Framework a MySQL

Entity Framework je ORM nástroj pro .NET. ORM slouží k práci s databázovými tabulkami jako s objekty. *Entity Framework* se stará hlavně o připojení k databázi, převod tabulek na objekty a jiné [2]. Díky tomu nemusíme implementovat datovou vrstvu a přístup k databázi. Typ databáze použité pro tento projekt je MySQL. Jedná se o poměrně rozšířený a oblíbený typ open-source databáze. *Entity Framework* je schopen díky adaptérům pracovat s různými typy databází. Konkrétně pro MySQL se jedná o *MySQL Connector .NET*. Práce s databází je více popsána v kapitole o aktualitách.

3.2.3 HTML a CSS

Pro stylování webu bylo použito klasické CSS. Rozhodoval jsem se, zda nepoužít modernější verze *Less* [3] nebo *Sass* [4], které podporují například proměnné, vnořování selektorů a jiné. Vytvořený kód se poté zkompiluje na klasické CSS. Oproti klasickému CSS nabízí větší přehled a lepší udržovatelnost kódu. Z důvodu mála zkušeností se jmenovanými technologiemi a poměrně malému rozsahu aplikace jsem zvolil použít klasické CSS. K referencím pro HTML a CSS jsem použil web *w3schools* [5].

3.2.4 GoogleMaps API

Obsahem webu je i mapa s lokací firmy. Z počátku jsem přemýšlel nad pouhým vložením obrázku mapy do stránky. Nicméně pro interkativní mapy *Google* vytvořil API, díky kterému je vložení funkční mapy jednoduché a rychlé. Jedná se o Javascript knihovnu, a s pomocí dokumentace [6] jsem byl schopen vytvořit skript pro požadovaný výsledek. Nejprve je nutno na požadovanou stránku přidat odkaz na API skript. Poté stačí vytvořit inicializační skript, ve kterém nastavíme vlastnosti mapy, jako například zakázaní kolečka na myši pro ovládání přiblížení a oddálení, nastavení typu mapy a hlavně středového bodu. Následuje vytvoření objektu pro samotnou mapu a případně značky. Ukázka inicializačního skriptu viz níže.

```
var mapCenter = new google.maps.LatLng(49.811621, 18.306077);

function initialize() {
  var mapProp = {
    center: mapCenter,
    zoom: 17,
    scrollwheel: false,
    mapTypeId: google.maps.MapTypeId.ROADMAP
  };
```

```
  var map = new google.maps.Map(document.getElementById("googleMap"),mapProp);

  var marker = new google.maps.Marker({
    position: mapCenter,
    });
  marker.setMap(map);
}

google.maps.event.addDomListener(window, 'load', initialize);
```

Výpis 1: Inicializační skript pro GoogleMaps

3.3 Jazykové verze

Web musel splňovat možnost přepnutí mezi anglickou a českou verzí. Ze začátku proběhla analýza možností řešení tohoto problému. V úvahu připadaly tři způsoby. Využití Javascript knihovny *i18next* [7], použití resource souborů v *.NET Framweworku* [8] nebo kompletní rozdělení prostoru pro českou a anglickou verzi. Vyzkoušel jsem si implementovat jak první způsob pomocí knihovny *i18next*, tak třetí způsob a to rozdělení prostoru.

3.3.1 Javascript knihovna i18next

Tato knihovna umožňuje mimo jiné funkce také lokalizaci webových stránek. Funguje na principu záměny hodnoty podle klíče ze slovníku. Použití této knihovny je rychlé a jednoduché. K HTML značce, jejíž obsah potřebujeme lokalizovat, stačí dopsat atribut *data-i18n="klíč"*. Poté je třeba vytvořit takzvané *locales* soubory. Tyto soubory jsou JSON formátu a obsahují dvojice klíč a hodnota. Pro každý jazyk existuje jeden takový soubor. Po načtení HTML stránky je spuštěn samotný *i18next* skript, který nahradí obsah všech HTML značek s daným atributem za hodnoty z *locales* sobuboru pro nastavený jazyk. Nastavit jazyk je možno více způsoby. Implicitně při spouštění skriptu, Javascript funkcí nebo URL parametrem. Výhody této možnosti jsou snadné použití a snadné přidání podpory pro další jazyk.

3.3.2 Rozdělení prostoru

Metoda rozdělení prostoru pro každý jazyk spočívá ve vytvoření kopie stávajících stránek a upravit je pro jiný jazyk. Na rozdíl od předešlých způsobů, tento nevyžaduje žádné další funkce ať už frameworku nebo Javascriptu. Stačí pouze vytvořit další HTML šablony. Nevýhodou této metody je ovšem větší náročnost na správu takovéto aplikace a udržování konzistentního vzhledu a informací. Tato náročnost narůstá s každým dalším podporovaným jazykem, proto není vhodná pro aplikace, které vyžadují podporu mnoha jazyků.

Po konzultaci s vedoucím práce, byl zvolen způsob rozdělení prostoru, z důvodu možnosti měnit ne jen hodnoty na jednotlivých stránkách, ale také jejich obsah a strukturu. V *ASP.NET Frameworku* jsem k tomuto řešení použil takzvané *areas*, tedy oblasti. Každá oblast má svůj Controller, Model i View, a dá se chápat jako podaplikace v aplikaci. Mezi praktické využití patří například administrátorská sekce aplikace nebo již zmíněné jiné jazykové prostředí. Pro tuto aplikaci tedy byla vytvořena nová oblast s názvem *en*, která obsahuje kompletně celou část webu v anglickém jazyce. Výchozí soubory mimo tuto oblast jsou určeny pro českou verzi. CSS a Javascript soubory jsou společné pro obě oblasti, ale je možné je podle potřeby rozdělit a využívat jiné. Pro přepínání do druhého jazyku slouží tlačítko v navigaci, které zjistí aktuální stánku, na které se nacházíme, a odkáže na tutéž, pouze pro jiný jazyk. Pokud jsme v anglické části aplikace, parametr *area* zůstane prázdný, tím řekneme, že se chceme dostat do výchozího adresáře. Naopak pokud se chceme přepnout do anglické verze, parametr *area* je nastaven na hodnotu *en*.

3.4 Responzivní design

Dalším z hlavních požadavků bylo vytvoření webu tak, aby se zobrazoval s ohledem na použité zařízení, například mobilní telefon a počítač. Toho lze z větší části dosáhnout pouze úpravami v CSS souboru a použitím *media queries*. *Media queries* je technika v CSS, která nám umožní aplikovat různé styly v závislosti na šířce obrazu. Použití *media queries* pro malou aplikaci je dostačující, nicméně pro responzivní a celkově *front-end* design existují různé frameworky. Ty umožňují využití CSS a Javasciript komponent pro jednodušší vývoj webu. Mezi nejznámější a nejvíce podporované komunitou patří *Bootstrap* [9] a *Foundation* [10]. Možnosti obou těchto frameworků jsou téměř totožné, a fungují velmi podobně. Po seznámení s oběma frameworky jsem se rozhodl využít framework *Bootstrap*, jelikož jsem s ním měl již nějaké zkušenosti a nabízel lepší práci s pozicováním. Porovnání frameworků a další informace jsem čerpal z webu *Responsive Design* [11].

```
<div class="container text-18">
  <div class="row">
    <!-- Pro mala zarizeni text zarovnany na stred, jinak vlevo -->
    <div id="addr" class="col-sm-6 col-md-4 col-lg-6 text-xs-center text-sm-
        left wow fadeIn">
      <h2 class="text-32 ct-comp-text">Adresa</h2>
      <!-- FontAwesome ikony -->
      <p><i class="fa fa-fw fa-map-marker ct-comp-icon"></i>Vratimovsk 624/11</
          p>
      <p><i class="fa fa-fw fa-location-arrow ct-comp-icon"></i>Ostrava,
          Ostrava-Kuniky</p>
      <p><i class="fa fa-fw fa-globe ct-comp-icon"></i>718 00</p>
```

```
    </div>
    <!-- Grid offset, pozicovani obrazku v gridu -->
    <div class="col-sm-5 col-sm-offset-1 col-md-4 col-md-offset-0 col-lg-3">
      <img class="card wow fadeIn" src="~/Content/Images/Bud.jpg" alt="Image 1"
           style="width:100%">
    </div>
    <div class="col-sm-5 col-sm-offset-7 col-md-4 col-md-offset-0 col-lg-3">
      <img class="card wow fadeIn" src="~/Content/Images/Man.jpg" alt="Image 2"
           style="width:100%">
    </div>
  </div>
</div>
```

Výpis 2: Ukázka definování různé šířky pro obsah v závislosti na velikosti displeje

Bootstrap pro řešení responzivního designu definuje takzvaný grid. To znamená, že oblast každého blokového prvku je rozdělena na sloupce, které mají pevnou šířku, a do těchto sloupců je možno pozicovat další obsah. Šířky sloupců jsou různé pro každou velikost obrazovky.

Bootstrap definuje 4 breakpointy pro *media queries*. Tyto body slouží k rozlišení šířky obrazovky a aplikovaní jiných stylů. Díky těmto bodům můžeme mít kontrolu nad tím, kolik prostoru zabere náš prvek na různě velkém displeji. Například pro malý displej obrázek zabírá celou šířku stránky, pro střední displej polovinu, a pro velký displej čtvrtinu sířky. Tohoto jsem využil a téměř každý prvek na stránce jsem upravil tak, aby se přizpůsobil velikosti obrazovky. Ukázka použití a nastavení jednotlivých prvků lze vidět na ukázce výše. Pro dodatečné úpravy jsem použil již zmíněné *media queries*. Jednalo se například o různé zarovnání textu, velikost písma, nebo nepovolení CSS animací pro nejmenší obrazovky. Ukázka použití CSS *media queries* je na výpisu níže.

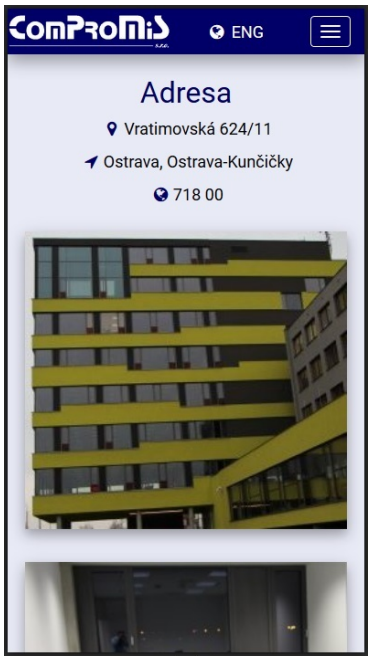
```
// Text zarovnany na stred pro male displeje
.text-xs-center {
    text-align: center;
}
// Animace pozadi v menu
.navslide {
    background-size: 100% 200%;
    background-image: linear-gradient(to bottom, #000068 50%, #ffffff 50%);
}
.navslide:hover {
    color: #000068 !important;
    background-color: #e8eaf6 !important;
    background-position: 0 100%;
```

```
}

// Tyto styly se aplikuji pouze pro obraz sirsi nez 768px
@media (min-width: 768px) {
    .text-sm-left {
        text-align: left;
    }
    // CSS animace nepovoleny pro male displeje
    .navslide {
        transition: all 0.3s ease-in 0s;
    }
}
```

Výpis 3: Ukázka použití CSS media queries

Pokročilejší změna obsahu, ke které nestačí pouze CSS, ale je zapotřebí i Javascript je například změna hlavního menu na mobilních telefonech. Na velkých displejích obsahuje hlavní navigace odkazy na jednotlivé stránky vedle sebe. Po zmenšení okna na určitou velikost se toto menu změní, a je zobrazeno pouze tlačítko pro rozbalení drop-down seznamu, ve kterém jsou jednotlivé položky. K tomuto je využita *Bootstrap* komponenta navigace. Na nejmenších displejích zabírá většina prvků celou šířku obrazovky, a prvky jsou tedy uspořádány pod sebou, jak lze vidět na obrázku 2.



Obrázek 2: Ukázka responzivního designu

3.5 Aktuality

Z počátku obsahoval web pouze jednu statickou aktualitu a do budoucna se počítalo s řešením, které bude aktuality stahovat a zobrazovat z databáze. K tomu byla potřeba vytvořit databázi a systém pro přidávání a správu těchto aktualit. Tvorba tohoto systému je popsána později v práci.

Po vytvoření systému vznikla i databáze, ve které jsou aktuality uloženy. Pro připojení k databázi a práci s ní je použit *Entity Framework*. *Entity Framework* umožňuje více způsobů pro práci s databází. Jedním z nich je *Database-First* přístup [12], který je využit v případě této aplikace. Tento přístup znamená, že *Entity Framework* automaticky vygeneruje model na základě existující databáze. Tyto třídy a databázový kontext, třídu zpracovávající připojení k databázi, poté můžeme použít v controlleru k získání dat a předání do view. V controlleru jak pro českou, tak anglickou verzi je obdobná metoda, jediný rozdíl je pouze ve výběru jazyka. Na výsledné stránce se zobrazí pouze ty příspěvky, které byly schváleny a vyhovují podmínkám zobrazení, jako jsou datum publikování a expirace. Metoda pro zobrazení jednoho příspěvku obsahuje pouze kontrolu parametru *id*. Poté se snaží z databáze získat příspěvek s tímto *id*. Pokud je příspěvek nalezen a vyhovuje podmínkám pro zobrazení, tak je tento objekt poslán do view. Pokud ne, tak proběhne přesměrování na seznam všech příspěvků. Zobrazení všech příspěvků má na starosti druhá metoda, která má navíc jako parametr číslo stránky. Příspěvky jsou stránkovány po určitém počtu a v případě chybného parametru je uživatel přesměrován opět na první stranu.

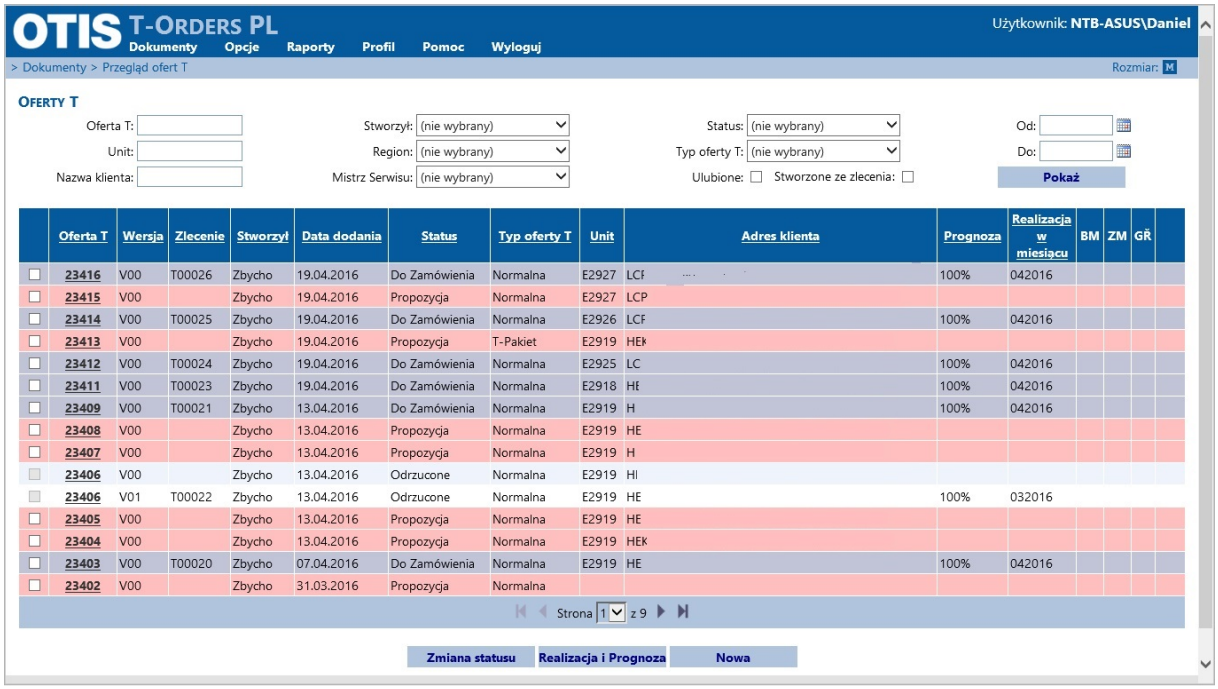
3.6 Řešení

Za použití výše zmíněných technologií a postupů, jsem vytvářel nové webové stránky pro firmu. Prvním krokem bylo vytvoření základní struktury stránek, takzvaný *layout*. *Layout* je šablona, která obsahuje části jako HTML hlavičku, ve které jsou připojeny odkazy na styly a případně skripty a další informace. Dále hlavní navigace a patička stránky. Mezi těmito částmi je definována oblast pro obsah, který rozšiřují další šablony. Úprava controlleru pro statický web je velmi jednoduchá a stačí pouze v metodě pro každou stránku zobrazit view. Metody pro aktuality jsou popsány výše. Pro překlad do angličtiny jsem nejdříve použil knihovnu *i18next*, ale po konzultaci jsem toto řešení nepoužil. Místo toho jsem vytvořil v aplikaci novou oblast pro anglickou část webu. Po vytvoření publikačního systému jsem doplnil aplikaci o model a připojení k databázi, ze které jsou stahovány aktuality. Web byl průběžně nahráván na firemní IIS server, na kterém jsem testoval nové verze. Časový odhad práce na tomto projektu je zhruba 17 dní. Práce bude dále pokračovat, ale již teď jsou tyto stránky v provozu a používány.

4 Aplikace TOrders

4.1 Zadání

Druhým úkolem ve firmě bylo seznámení se s aplikací *TOrders* a pomoc při vytváření reportů. Aplikace *TOrders* je vyvíjená pro firmu OTIS, která ji využívá pro menší, takzvané T zakázky. Jde například o výměnu dveří nebo panelu v kabině výtahu a jiné. Aplikace je nasazena a využívána ve více zemích po Evropě. Já konkrétně jsem pracoval s verzemi pro Bulharsko a Polsko. Mým úkolem bylo v každé z těchto aplikací vytvořit reporty pro výstup z jednotlivých zakázek, dalo by se říci, že jde o styl faktury. Na reportech jsou informace o zakázce, kontaktní údaje, přehled použitého materiálu a další doplňující údaje. Prostředí aplikace je na obrázku 3.



Obrázek 3: Prostředí aplikace TOrders

4.2 Použité technologie

4.2.1 Client Report Definition

Report je grafické zpracování dat. *Client Report Definition (.rdlc)* jsou soubory, které popisují jak má výsledný report vypadat a jaká použije data. Ke každému reportu může být přiřazen *data set*, výstup z databáze s požadovanými daty, se kterými chceme v reportu pracovat. Reporty jsou vytvářeny v grafickém návrháři ve *Visual Studiu*. Jako obsah reportu mohou být různé objekty, jako třeba tabulky, obrázky, grafy a jiné komponenty. Výhodou a usnadněním práce je možnost vkládání reportů do sebe, takzvaný *subreport*. Důležitou součástí je *ReportViewer*, komponenta, která se stará o zobrazení reportu na webové stránce nebo formuláři. *ReportViewer*

také umožňuje další manipulaci s výsledným dokumentem jako tisk, nebo konverze například do formátu PDF či *Microsoft Word*. Právě tyto konverze byly důležité, a je potřeba, aby výsledek vypadal co nejvíce podobně v každém formátu.

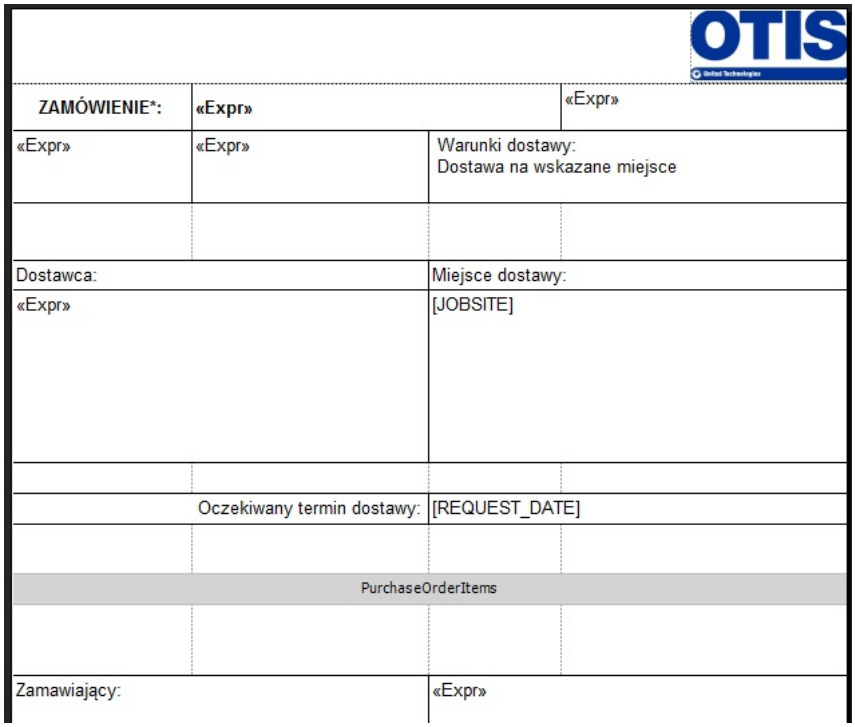
4.2.2 Subversion

Ve firmě je většina zdrojových kódů uložena na serveru. K jejich správě je využit *Subversion*. *Subversion* je open-source verzovací systém pro zdrojové kódy. Jako ostatní systémy, i *Subversion* nabízí standartní funkce jako commit, rozdělení do větví, spojování větví a jiné [13]. Rozdíl oproti například populárnímu systému *Git* je ten, že *Subversion* je centralizovaný. To znamená, že existuje jeden repozitář na serveru a uživatelé pracují s ním, kdežto v případě systému *Git* má každý uživatel lokální kopii repozitáře a může pracovat i bez nutnosti připojení k serveru.

4.3 Postup řešení

Nejprve ze všeho jsem musel získat zdrojové kódy aplikací, ty jsou, jak už jsem popsal uloženy v *Subversion* repozitáři na firemním serveru. K práci se systémem *Subversion* jsem využil *AnkhSVN* [19]. Jedná se o doplněk pro *Visual Studio*, který přidává podporu *Subversion* přímo do vývojového prostředí. Po nainstalování doplňku *AnkhSVN* je možno otevřít projekt ve *Visual Studiu* přímo ze zadaného repozitáře, a tím se celý projekt stáhne. Aplikace využívá databázi Oracle. Kvůli tomu bylo nutné stáhnout *Oracle Data Access (ODAC)*, knihovny potřebné pro připojení k této databázi, a přidat na ně reference. Po spuštění aplikace a vytvoření uživatelského účtu jsem mohl prozkoumat nejen kódy, ale i funkčnost. Hlavní zaměření bylo na stránku zobrazující reporty, a poté na samotné již existující reporty.

Výsledná podoba reportů se měla co nejvíce přiblížit zadání od zákazníka, a to jak při zobrazení na webu, tak při exportu do PDF. Právě při různých zobrazení vzniká problém s formátováním jako odsazení a okraje. Řešením jak tyto odchylky co nejvíce eliminovat je vložení tabulky přes celé tělo reportu. Tím dosáhneme celkem velké kontroly nad velikostí jednotlivých řádků. Navíc spojením buněk můžeme jednoduše určit šířku řádku. Další výhodou tabulky je jednodušší práce s *data setem*. Tabulce lze přidělit *data set*, a pro zobrazení hodnoty stačí akorát napsat jednoduchý výraz s názvem atributu. Důležitými body při tvorbě reportů byl odpovídající formát podle zadání od zákazníka a správné použití dat. Návrh reportu ve *Visual Studiu* je na obrázku 4.



Obrázek 4: Návrh vzhledu reportu

Každý report může také obsahovat hlavičku a patičku, která se opakuje na každé stránce, pokud se nejedná pouze o jednostránkový report. Podle zadání jsem vytvořil několik reportů jak pro bulharskou, tak polskou verzi aplikace. Většina dat, která jsem potřeboval pro tyto reporty, již byla vytvořena. Pokud byla potřeba v datových zdrojích něco změnit, konzultoval jsem tyto změny s kolegou, a po provedených změnách jsem si aplikaci aktualizoval na nejnovější verzi. V případě, že se report skládal z více částí, jako seznam použitého materiálu či přílohy, tak byly tyto části vytvořeny jako další report, který obsahoval pouze tento seznam a následně byl vložen do hlavního reportu. Důvod je ten, že tyto data jsou v jiných tabulkách a je proto nutno použít jiný zdroj. Návrhář pro reporty je podobný jako například u *Windows Forms* aplikací a dovoluje hodně možností formátování. V obou aplikacích jsem průběžně vytvořil několik reportů a subreportů a práce na tomto úkolu zabrala zhruba 6 dní. Vyzkoušel jsem si hlavně práci se systémem *Subversion* a seznámil se zpracováním dat pomocí reportů.

5 Publikační systém

5.1 Zadání

Třetím úkolem ve firmě bylo vytvoření publikačního systému. Tento systém má sloužit k přidávání, úpravě a kontrole obsahu. Z počátku se jedná pouze o správu aktualit, konkrétně pro firemní web. Rozšířená verze aplikace bude použita i pro další projekty, které se budou ve firmě vytvářet. Publikační systém měl umožnit správu obsahu na základě schvalovacího procesu, a ne pouhým přidáním. K tomu jsou potřeba uživatelské role a jiná přístupová práva. Běžný uživatel může přidávat a upravovat své příspěvky, zatímco pro přidání obsahu na web je potřeba schválení. Požadavek byl na vytvoření a práci s MySQL databází, která bude vytvořena na firemním serveru.

5.2 Použité technologie

5.2.1 ASP .NET Framework

Jako u prvního projektu, i tato aplikace je vyvíjena s pomocí *ASP.NET Frameworku*, opět MVC přístup. Důvody k použití tohoto frameworku jsou popsány u prvního úkolu.

5.2.2 Entity Framework a MySQL

Jako úložiště dat zde byla vyžadována MySQL databáze. K přístupu a práci s touto databází jsem opět použil *Entity Framework*, v tomto případě takzvaný *Code-First* přístup [18]. Tento přístup znamená, že databáze je vytvářena a upravována podle toho, jak vypadají naše modelové třídy. Je zde využit systém migrací. Migrace jsou soubory obsahující metody, které popisují, jak se má upravit databáze v případě aplikování migrace, ale také jaké změny se mají provést při odstranění této migrace. Použití MySQL databáze s *Entity Frameworkem* vyžaduje dodatečné nastavení [21]. Nejdůležitější je stáhnout a přidat reference na *MySQL Connector .NET*. Jedná se o knihovny, které umožní *Entity Frameworku* spolupracovat s MySQL databází.

5.2.3 HTML a CSS

Pro stylování webu jsem použil opět klasické CSS v kombinaci s frameworkem *Bootstrap*. Styl aplikace je obdobný jako u prvního projektu a je ve stejných barvách. U této aplikace nebyl kladen větší důraz na responzivní design, jelikož se jedná především o aplikaci určenou pro klasické desktopy. HTML šablony jsou zaměřeny především na formuláře a stránky zobrazující informace o modelu. Z toho důvodu je použito mnohem více *Razor* syntaxe. Tabulky, tlačítka a možnosti v menu jsou doplněny o ikony opět za použití CSS *Font Awesome* [20]. Písmo použité na webu je stejné jako v prvním projektu, a to písmo Roboto.

5.2.4 Javascript

V projektu je použito několik Javascript knihoven pro přidání funkcionality do aplikace. Základem pro tyto knihovny je *jQuery*. Na *jQuery* je závislé mnoho dalších knihoven, a v dnešních aplikacích se *jQuery* vyskytuje téměř vždy. Samotný *Bootstrap* jej vyžaduje pro správnou funkčnost svých komponent.

Prvním skriptem je *Datepicker*. *Datepicker* nám umožní rozbalit okno pro výběr data, při vkládání do formuláře. Použití tohoto skriptu je jednoduché, ale při jeho inicializaci máme mnoho možností jak chování skriptu ovlivnit [14]. Tento skript je používán při výběru data publikace a expirace při schvalovacím procesu příspěvku.

Dalším velmi užitečným skriptem je *Bootstrap Table*. Jedná se o, jak už název napovídá, rozšíření základních HTML tabulek a přidání mnoha funkcí. Nastavení probíhá pro každou tabulku zvlášť a máme na výběr z různých možností a atributů, jak chceme, aby se tabulka rozšířila a jaké podporovala funkce [15]. Například stránkování, seskupování sloupců, detailní pohled, editování přímo v tabulce a jiné.

Posledním použitým skriptem je *TinyMCE*. Jedná se o skript, který změní formulářový vstup *<textarea>* na plnohodnotný textový editor. Použití skriptu je opět velmi jednoduché a jeho nastavení probíhá při inicializaci, podobně jako u skriptu *Datepicker*. Pokud nechceme zaměnit všechny vstupy *<textarea>*, je možné při inicializaci nastavit selektor tak, aby byly upraveny pouze vstupy se specifickým ID nebo třídou. Editor je možno rozšířit o mnoho modulů [16] a také nastavit jak chceme, aby editor vypadal, a jaké funkce umožňoval.

5.3 Analýza a Konfigurace

Nejprve ze všeho byla provedena analýza zadání a konzultace s vedoucím praxe, pro získání dodatečných informací o aplikaci. Tvorba projektu byla průběžně konzultována, tak aby projekt odpovídal požadavkům. Po dohodnutí na zvolených technologiích a způsobu řešení následoval další krok, vytvoření projektu a základní konfigurace.

Po vytvoření nového projektu ve *Visual Studiu*, typu ASP.NET MVC aplikace byla potřeba přidat reference na knihovny pro práci s MySQL. Poté jsem přidal ke třídě *ApplicationDbContext* atribut *[DbConfigurationType(typeof(MySqlEFConfiguration))]*. Tato třída se stará o komunikaci mezi Entity Frameworkem a databází. Obsahuje důležité třídní členy typu *DbSet*, například *DbSet<Image>*. Díky této kolekci můžeme pracovat s obsahem databáze jako s objekty v aplikaci. Dalším krokem byla úprava souboru *Web.config*. Bylo zde přidána reference na MySQL klienta a řetězec pro připojení k databázi.

Po základní konfiguraci aplikace jsem vytvořil novou databázi na lokálním MySQL serveru. K přístupu a správě této databáze jsem využil nástroj *MySQL Workbench*. Jelikož se o strukturu databáze stará *Entity Framework* a systém migrací, stačilo pouze vytvořit nové schéma.

5.4 Tvorba modelu

Dalším krokem bylo vytvoření třídy *Post*. Tato třída je model, který obsahuje všechna potřebná data pro příspěvek. V modelu jsou uvedeny i odkazy na cizí klíče, ty jsou označeny klíčovým slovem *virtual*. Díky tomu atributy podporují *Lazy Load*, metodu která načítá obsah těchto atributů až v momentě, kdy jsou tyto informace potřeba. Jednotlivé členy třídy lze označit atributy, které přidávají další informace pro *ASP.NET*, nebo *Entity Framework*. Mezi tyto atributy patří například *[Key]*, pro označení primárního klíče, *[NotMapped]*, pro určení, že tento člen má být ignorován *Entity Frameworkem*, *[Required]*, pro označení povinnosti, anebo *[DisplayFormat()]*, například pro určení formátu data.

V průběhu vývoje vznikly i další modely pro potřebu aplikace. Některé jsou použity pouze pro view, jako například *UsersViewModel*, který obsahuje informace základní informace o uživateli a kolekci rolí do kterých je přiřazen. Modely, které upravují strukturu databáze, jsou další dva, a to *Image*, obsahující informace o obrázku, a *Message*, který obsahuje informace o zprávě připojené k příspěvku.

Po každé úpravě některého z modelů, které zasahují do databáze, je potřeba vytvořit novou migraci a aplikovat ji. Novou migraci vytvoříme v příkazovém řádku Visual Studia příkazem *Add-Migration „název“*. Tento příkaz vygeneruje nový migrační soubor se změnami v modelu oproti aktuálnímu stavu databáze. Aktualizaci databáze a aplikování migrací provedeme příkazem *Update-Database*. Ukázka migrace je na výpisu kódu níže.

```
public partial class PostImage : DbMigration
{
    public override void Up()
    {
        CreateTable(
            "dbo.Images",
            c => new
                {
                    Id = c.Int(nullable: false, identity: true),
                    Title = c.Int(nullable: false),
                    Alt = c.Int(nullable: false),
                    Url = c.String(unicode: false),
                })
            .PrimaryKey(t => t.Id);

        AddColumn("dbo.Posts", "Image_Id", c => c.Int());
        CreateIndex("dbo.Posts", "Image_Id");
        AddForeignKey("dbo.Posts", "Image_Id", "dbo.Images", "Id");
    }
```

```
    public override void Down()
    {
        DropForeignKey("dbo.Posts", "Image_Id", "dbo.Images");
        DropIndex("dbo.Posts", new[] { "Image_Id" });
        DropColumn("dbo.Posts", "Image_Id");
        DropTable("dbo.Images");
    }
}
```

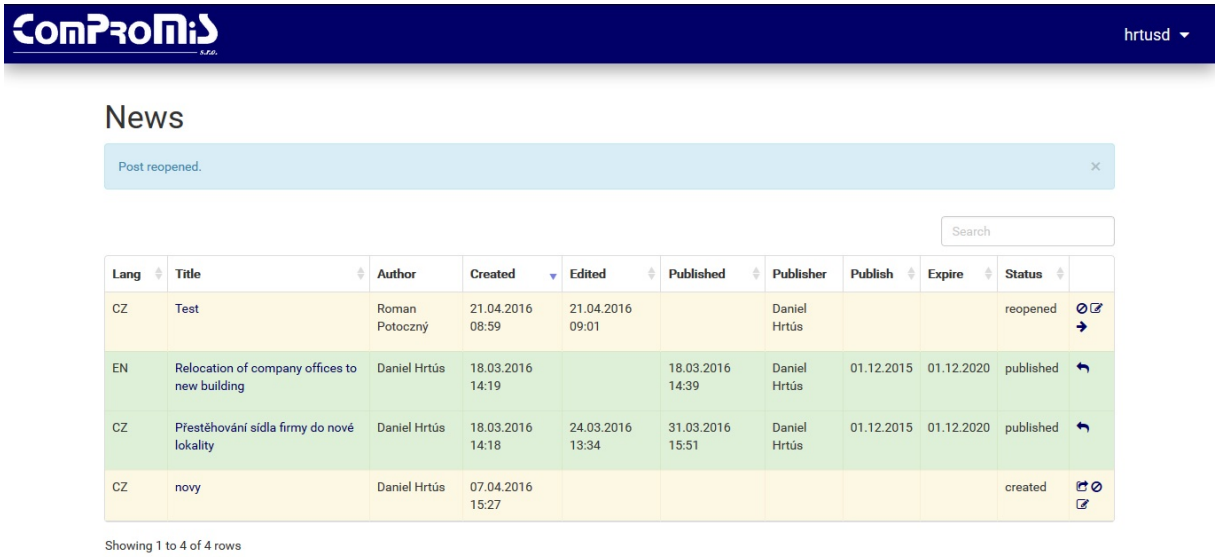
Výpis 4: Migrační metoda

5.5 Prostředí aplikace

Dalším krokem bylo vytvoření základních šablon s formuláři pro přidání, editaci a detail příspěvku. V *ASP.NET* existují pomocné metody pro vytváření HTML obsahu. Mezi některé z nich patří vytvoření formuláře, či přidání pole pro zadání hodnot ve formuláři. Tyto metody jsou například *Html.BeginForm()* a *Html.DisplayFor()*.

V šablonách pro přidání a editaci příspěvku je použit editor *TinyMCE* pro úpravu obsahu příspěvku. Výstup z tohoto editoru je poté uložen do databáze jako HTML řetězec, a při zobrazení tohoto řetězce jsou HTML značky použity pro formátování obsahu. Díky tomu je u obsahu příspěvku možno změnit barvu písma, zarovnání, vytvořit seznam, nadpis a jiné.

Hlavním oknem po přihlášení je tabulka se seznamem všech příspěvků. Tato tabulka je rozšířena skriptem *Bootstrap Table* a umožňuje řazení podle jednotlivých sloupců, stránkování obsahu, a nebo také filtraci podle klíčového slova. U každého z příspěvku jsou možnosti dostupné pro daného uživatele. Detailní informace se zobrazí po kliknutí na titulek příspěvku. Navigace v aplikaci je řešena jednoduchým rozbalovacím menu v pravém horním rohu. Obsah menu se mění podle rolí uživatele. Hlavní okno aplikace je na obrázku 5.



Obrázek 5: Hlavní okno aplikace

5.6 Logika aplikace

Pokud máme připravenou databázi, modelové třídy a základní šablony pro zpracování dat, můžeme začít s tvorbou logiky aplikace. Vytvořil jsem tedy třídu *PostController* a *RolesController*. Třída *PostController* obsahuje metody pro všechny operace s příspěvky, jako například vytvoření, editace, publikování a další. *RolesController* řeší práci s rolemi a uživateli. Každá metoda odpovídá jednomu požadavku například typu GET nebo POST. Metody se také dají označit atributy, například atributem *[HttpPost]*, ten značí, že se jedná o požadavek typu POST. Atributem *[Authorize(Roles = „role“)]* můžeme zamezit přístupu uživatelům, kteří nemají dostatečná práva pro tuto operaci.

K autentizaci, autorizaci a celkové správě uživatelských účtů jsou využity předvytvořené třídy a metody, které se nazývají *ASP.NET Identity*. Cílem tohoto systému bylo sjednotit identitní systém pro všechny *ASP.NET* platformy, jako například MVC, Web Forms, SignalR a další. Umožňuje jednoduché vytváření a přiřazování do rolí, podporuje přihlašování přes různé extérní služby jako například Facebook, Twitter, Google a další [17].

Třída pro uživatele obsahuje pouze základní atributy jako email, uživatelské jméno, heslo atd. Tuto třídu jsem rozšířil o atributy jméno a příjmení. V aplikaci jsou rozlišovány tři uživatelské role, které určují práva a možnosti pro každého uživatele. Pro práci s rolemi bylo potřeba vytvořit metody pro přidání a odstranění uživatele z role. Metoda pro odstranění uživatele z role je na ukázce. Role v aplikaci jsou následující:

- User - umožňuje procházet všechny příspěvky, přidávat nové příspěvky, editovat ty, které uživatel vytvořil a odeslání příspěvku k publikaci.

- Publisher - má práva ke schválení příspěvků, které jsou označeny jako připravené, případně jejich zamítnutí.
- Admin - umožňuje pouze správu uživatelů, přiřazování do rolí a smazání příspěvku.

```
// GET: Roles/DeleteRoleForUser
public ActionResult DeleteRoleForUser(string UserName, string RoleName)
{
    if (UserName.Equals("admin", StringComparison.CurrentCultureIgnoreCase) &&
        RoleName == "Admin")
    {
        TempData["res"] = "Cannot remove user admin from Admin role.";
        return Redirect(Request.UrlReferrer.ToString());
    }

    var user = db.Users.Where(u => u.UserName.Equals(UserName, StringComparison
        .CurrentCultureIgnoreCase)).FirstOrDefault();

    if (UserManager.IsInRole(user.Id, RoleName))
    {
        UserManager.RemoveFromRole(user.Id, RoleName);
        TempData["res"] = "Role removed from this user successfully.";
    }
    else
    {
        TempData["res"] = "This user doesn't belong to selected role.";
    }

    ViewBag.Roles = db.Roles.OrderBy(r => r.Name).ToList().Select(rr => new
        SelectListItem { Value = rr.Name.ToString(), Text = rr.Name }).ToList();

    return Redirect(Request.UrlReferrer.ToString());
}
```

Výpis 5: Metoda pro smazání uživatele z role

Hlavním bodem, kterým se aplikace zabývá je příspěvek a jeho stav. Každý příspěvek má status, který určuje, v jakém je stavu a jaké operace s ním je možno provádět. Stavů je celkem pět a navíc také určují, kdo s příspěvkem může manipulovat.

- Created – výchozí stav po vytvoření nového příspěvku, autor může příspěvek editovat, zavřít nebo odeslat ke schválení.

- Ready – příspěvek byl odeslán ke schválení, autor tento příspěvek může stáhnout zpět, pokud objevil chybu, schvalovatel může buď příspěvek publikovat, nebo poslat zpět k autorovi s volitelnou zprávou, například s důvodem zamítnutí.
- Published – publikovaný příspěvek, autor už za něj nezodpovídá, schvalovatel jej může stáhnout zpět do stavu reopened.
- Reopened – stav kdy schvalovatel může příspěvek editovat a poté znovu publikovat, případně zavřít.
- Closed – zavřený příspěvek, není možno s ním dál pracovat.

Každá změna stavu příspěvku přidá do databáze zprávu o tom, kdy tato změna nastala, kdo ji provedl a také připojený text. Historie příspěvku je poté zobrazena na detailu každého příspěvku, spolu s dalšími informacemi. Do této historie je také kromě automatických správ přidat zprávu při vytváření a editaci příspěvku, například důvod. Stejnou možnost má schvalovatel v případě, že příspěvek jako označený ke schválení zamítne a odešle zpět k autorovi.

5.7 Dokončení

Poté, co aplikace byla připravena k prvnímu testování na serveru, bylo potřeba aplikaci nahrát na server. K tomu mi byl zřízen uživatelský účet na firemním IIS serveru. Dále jsem potřeboval účet i pro práci s databází v aplikaci *phpMyAdmin*. Na firemním MySQL serveru jsem vytvořil přesnou kopii databáze jako na lokálním serveru při vývoji a testování aplikace. Po prvním spuštění aplikace se provede metoda, která vytvoří uživatelské role a výchozí administrátorský účet, pokud neexistují. Tímto je zajištěna možnost aplikaci používat bez dalšího zásahu administrátora. Tato metoda je na výpisu kódu níže.

```
public static void SeedDb()
{
    var roleManager = new RoleManager<IdentityRole>(new RoleStore<IdentityRole
        >(new ApplicationDbContext()));
    var userManager = new UserManager<ApplicationUser>(new UserStore<
        ApplicationUser>(new ApplicationDbContext()));

    if (roleManager.FindByName("Admin") == null)
    {
        roleManager.Create(new IdentityRole { Name = "Admin" });
    }
    if (roleManager.FindByName("Publisher") == null)
    {
        roleManager.Create(new IdentityRole { Name = "Publisher" });
```

```
    }
    if (roleManager.FindByName("User") == null)
    {
        roleManager.Create(new IdentityRole { Name = "User" });
    }

    if (userManager.FindByName("admin") == null)
    {
        var user = new ApplicationUser { UserName = "admin", Email = "
            admin@admin.com", FirstName = "admin", LastName = "admin" };
        var result = userManager.Create(user, "***************"); // defaultni
            heslo
        if (result.Succeeded == false)
        {
            throw new Exception(result.Errors.First());
        }
        userManager.AddToRole(user.Id, "Admin");
    }
}
```

Výpis 6: Metoda pro vytvoření uživatelských rolí a administrátorského účtu

Požadavky na aplikaci se v průběhu vývoje měnily, a proto častěji probíhalo nasazení na server a testování nových fukncí. Odhadovaný čas strávený na tomto projektu je zhruba 20 dní. Projekt je v současné době používán pro správu aktualit na novém firemním webu. Nicméně práce dále pokračuje a projekt bude dále rozšiřován, aby mohl být využit v dalších aplikacích.

6 Závěr

Absolvování odborné praxe ve firmě ComProMiS s.r.o. bych hodnotil velmi kladně. Rozšířil jsem si praktické i teoretické znalosti v dané problematice, vyzkoušel si spolupráci na existující aplikaci a průběh aplikace od analýzy a návrhu až po nasazení na server a spuštění. Při absolvování odborné praxe ve firmě jsem uplatnil znalosti především z předmětů Programovací jazyky II, Architektura .NET a také Vývoj Informačních Systémů. Konkrétně se jedná o programování v jazyku C# pro platformu *.NET*. Znalosti které mi scházely byly vytváření webových aplikací pomocí *ASP.NET* a práce s *Entity Frameworkem*.

Výsledkem prvního zadaného projektu jsou nové webové stránky pro firmu. Při práci na tomto projektu jsem využil *ASP.NET Framework* a architekturu MVC. Zobrazované aktuality jsou stahovány z MySQL databáze. Pro připojení k této databázi a práci s daty je použit *Entity Framework*, konkrétně *Database-First* přístup. Responzivní design webu je řešen s pomocí front-end frameworku *Bootstrap* a použitím *media queries* v CSS.

V druhém úkolu jsem se seznámil s, pro mne, novou technologií reportů. Vytvořil jsem několik reportů pro aplikaci *TOrders*, které jsou používány jak v polské, tak bulharské verzi. Přínosem pro mne bylo seznámení s verzovacím systémem *Subversion* a jeho prací ve *Visual Studiu*, důležité ke spolupráci na vývoji aplikace s ostatními kolegy.

Jako poslední vznikl projekt, který umožnuje správu příspěvků. Stejně jako v prvním projektu, i zde byla pro vývoj použita architektura MVC. Nicméně je zde kladen větší důraz na logiku aplikace, než na prezentační vrstvu. Vyzkoušel jsem si další přístup pro práci s databází, a to takzvaný *Code-First*. Díky tomu jsem se setkal s migracemi, postupným vytvářením a úpravou databáze.

Oba projekty co jsem vytvořil, byly také nasazeny na firemní IIS server a MySQL server. Během vývoje jsem se seznámil s mnoha různými frameworky nebo moduly, ať už se jedná o frameworky jako *Foundation* nebo *Bootstrap*, tak Javascript knihovny jako *i18next* či *TinyMCE*. Velkým přínosem pro mne bylo vytvoření reálných aplikací. Jak webové stránky firmy, tak publikační systém budou postupně rozšiřovány a upravovány, ale již teď jsou obě tyto aplikace firmou používány. Věřím, že získané zkušenosti pro mne budou užitečné ať už při dalším studiu, tak při práci v daném oboru.

Literatura

[1] *ComProMiS s.r.o.* [online]. [cit. 2016-04-20]. Dostupné z: http://www.compromis.cz/

[2] *Entity Framework (EF) Documentation* [online]. [cit. 2016-04-20]. Dostupné z: https://msdn.microsoft.com/en-us/data/ee712907

[3] *Less* [online]. [cit. 2016-04-20]. Dostupné z: http://lesscss.org/

[4] *Sass* [online]. [cit. 2016-04-20]. Dostupné z: http://sass-lang.com/

[5] *W3schools* [online]. [cit. 2016-04-20]. Dostupné z: http://www.w3schools.com/

[6] *Google Maps JavaScript API* [online]. [cit. 2016-04-20]. Dostupné z: https://developers.google.com/maps/documentation/javascript/

[7] *i18next: internationalization framework* [online]. [cit. 2016-04-20]. Dostupné z: http://i18next.com/

[8] *Globalization And Localization With Razor Web Pages* [online]. [cit. 2016-04-20]. Dostupné z: http://www.mikesdotnetting.com/article/183/globalization-and-localization-with-razor-web-pages

[9] *Bootstrap* [online]. [cit. 2016-04-20]. Dostupné z: http://getbootstrap.com/

[10] *Foundation* [online]. [cit. 2016-04-20]. Dostupné z: http://foundation.zurb.com/

[11] *Responsive Web Design* [online]. [cit. 2016-04-20]. Dostupné z: https://responsivedesign.is/

[12] *EF Database First with ASP.NET MVC: Creating the Web Application and Data Models* [online]. [cit. 2016-04-20]. Dostupné z: https://www.asp.net/mvc/overview/getting-started/database-first-development/creating-the-web-application

[13] *Apache Subversion* [online]. [cit. 2016-04-20]. Dostupné z: https://subversion.apache.org/

[14] *Bootstrap-datepicker* [online]. [cit. 2016-04-20]. Dostupné z: https://bootstrap-datepicker.readthedocs.org/en/latest/

[15] *Bootstrap Table* [online]. [cit. 2016-04-20]. Dostupné z: https://github.com/wenzhixin/bootstrap-table

[16] *TinyMCE Documentation* [online]. [cit. 2016-04-20]. Dostupné z: https://www.tinymce.com/docs/

[17] *Introduction to ASP.NET Identity* [online]. [cit. 2016-04-20]. Dostupné z: http://www.asp.net/identity/overview/getting-started/introduction-to-aspnet-identity

[18] *Code First Migrations and Deployment with the Entity Framework in an ASP.NET MVC Application* [online]. [cit. 2016-04-20]. Dostupné z: http://www.asp.net/mvc/overview/getting-started/getting-started-with-ef-using-mvc/migrations-and-deployment-with-the-entity-framework-in-an-asp-net-mvc-application

[19] *AnkhSVN* [online]. [cit. 2016-04-20]. Dostupné z: https://ankhsvn.open.collab.net/

[20] *Font Awesome* [online]. [cit. 2016-04-20]. Dostupné z: http://fontawesome.io/

[21] *ASP.NET: Setup a MVC5 website with MySQL, Entity Framework 6 Code-First and VS2013* [online]. [cit. 2016-04-20]. Dostupné z: http://www.ryadel.com/en/asp-net-setup-mvc5-website-mysql-entity-framework-6-code-first-vs2013/